# bfGWAS: Bayesian Functional GWAS

Jingjing Yang

February 3, 2017

## Contents

# 1 Introduction

This software (bfGWAS) implements a Bayesian hierarchical model for genome-wide association studies (GWASs), which integrates genomic annotations into a Bayesian variable selection regression (BVSR) model. The key features of bfGWAS are accounting for linkage disequilibrium (LD) and functional genomic information, capable of identifying multiple associations per locus, and scalable for large GWAS with a novel computation algorithm — Expectation-Maximization Markov chain Monte Carlo (EM-MCMC). Compared to another Bayesian GWAS software GEMMA [5] (`http://www.xzlab.org/software.html`) based on the standard BVSR model, bfGWAS has the advantages of memory efficiency (saves up to 97% memory usage) and good MCMC convergence property (i.e., greatly improves the mixing property for posterior samples).

## 1.1 BVSR with genomic annotations

### 1.1.1 Standard BVSR

Consider the following standard BVSR model

$$\boldsymbol{y}_{n\times 1} = \boldsymbol{X}_{n\times p}\boldsymbol{\beta}_{p\times 1} + \boldsymbol{\epsilon}_{n\times 1},\ \epsilon_i \sim N(0,\tau^{-1}),\ \beta_i \sim \pi_i N(0,\tau^{-1}\sigma_i^2) + (1-\pi_i)\delta_0(\beta_i), \tag{1}$$

where $n$ denotes the number of individuals; $p$ denotes the number of genetic variants; $\boldsymbol{y}_{n\times 1}$ denotes the phenotype vector; $\boldsymbol{X}_{n\times p}$ denotes the genotype matrix; $\epsilon_i$ is the residual error independently and identically distributed (i.i.d.) with the normal distribution with mean 0 and variance $\tau^{-1}$, $N(0,\tau^{-1})$; and $\beta_i$ follows a point-normal prior (also known as the spike-and-slab prior) distribution [2, 3, 4] — that is, $\beta_i$ independently follows the normal distribution $N(0,\tau^{-1}\sigma_i^2)$ with probability $\pi_i$ and the point mass function at 0, $\delta_0(\cdot)$, with probability $(1-\pi_i)$ ($\delta_0(\beta_i)=1$ if $\beta_i=0$, otherwise $\delta_0(\beta_i)=0$).

The genotype matrix $\boldsymbol{X}_{n\times p}$ contains either dosage data within range $[0,2]$ or genotyped data with values $\{0,1,2\}$ denoting the number of minor alleles; the assumption of the point normal prior for $\beta_i$ enforces variable selection in the regression model (1); and the intercept term is dropped here because we assume both $\boldsymbol{y}_{n\times 1}$ (corrected for covariates) and columns of $\boldsymbol{X}_{n\times p}$ are centered. We further assume $\boldsymbol{y}_{n\times 1}$ is standardized with variance 1. Although this model is developed for quantitative trait, it can be applied on case-control studies by quantifying cases and controls as 1's and 0's following previous approaches [3, 4].

### 1.1.2 Integrating Genomic Annotations

For now, bfGWAS considers only non-overlapped binary annotations ($Q > 1$). Let $\boldsymbol{A_i} = (A_{i1},\cdots,A_{iQ})^T$ denotes the vector of annotations for the $i$th variant, and $A_{iq}$ takes binary values (1/0) to denote with or without a particular feature. In order to integrate genomic annotations in

the standard BVSR model (1), we assume $\pi_i = \pi_q$, $\sigma_i^2 = \sigma_q^2$ for all variants that are of annotation $q$, implying the same point-normal prior for all variants of the same annotation. We further assume the following priors (independent of each other):

$$\pi_q \text{ i.i.d.} \sim Beta(a_q, b_q),\ \sigma_q^2 \text{ i.i.d.} \sim IG(k_3, k_4),\ \tau \sim G(k_1, k_2), \tag{2}$$

where $(a_q, b_q)$ in the Beta distribution could be different with respect to different annotations, $IG(k_1, k_2)$ denotes an Inverse-Gamma distribution with shape parameter $k_1$ and scale parameter $k_2$, and $G(k_3, k_4)$ denotes a Gamma distribution with shape parameter $k_3$ and scale parameter $k_4$. Note that this BVSR model collapses to the standard BVSR model if all variants are of the same annotation ($Q = 1$).

### 1.2 Latent Indicator Variable

To facilitate computation, a latent indicator vector $\boldsymbol{\gamma}_{p\times 1}$ [2] is introduced into the model, where each element $\gamma_i \in \{0, 1\}$ indicates whether the corresponding $i$th effect $\beta_i$ equals to 0 ($\gamma_i = 0$) or follows the $N(0, \tau^{-1}\sigma_i^2)$ distribution ($\gamma_i = 1$). Equivalently,

$$\gamma_i \sim Bernoulli(\pi_i),\ \boldsymbol{\beta}_{-\boldsymbol{\gamma}} \sim \delta_0,\ \boldsymbol{\beta}_{\boldsymbol{\gamma}} \sim MVN_{|\boldsymbol{\gamma}|}(0, \tau^{-1}\boldsymbol{V}_{\boldsymbol{\gamma}}),$$

where $|\boldsymbol{\gamma}|$ denotes the number of non-zero entries in $\boldsymbol{\gamma}$; $\boldsymbol{\beta}_{-\boldsymbol{\gamma}}$ denotes the sub-vector of $\boldsymbol{\beta}_{p\times 1}$ corresponding to variants with $\gamma_i = 0$; $\boldsymbol{\beta}_{\boldsymbol{\gamma}}$ denotes the sub-vector of $\boldsymbol{\beta}_{p\times 1}$ corresponding to the variants with $\{\gamma_j = 1; j = 1, \cdots, |\boldsymbol{\gamma}|\}$; and $\boldsymbol{V}_{|\boldsymbol{\gamma}|}$ is the corresponding sub-matrix (for all $\gamma_j = 1$) of $\boldsymbol{V}_{p\times p}$ whose $i$th diagonal element is $V_{ii} = \sigma_i^2$.

### 1.3 EM-MCMC Algorithm

First of all, assume the genotype data $\boldsymbol{X}$ is segmented into $K$ blocks, i.e., $\boldsymbol{X} = \{\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_K\}$, where each sub-matrix $\boldsymbol{X}_k$ has dimension $n \times p_k$ (genotypes of $p_k$ variants for $n$ samples). It is highly recommended to segment blocks with about $5,000 \sim 10,000$ variants. The outline of the EM-MCMC algorithm is stated as follows (details are available in the Supplement of the main paper):

(i) Fix $\tau$ value as the phenotype variance (will not be updated);

(ii) Set initial values for the category specific parameters $(\boldsymbol{\pi}, \boldsymbol{\sigma^2})$;

(iii) E-step: Within each genome block, implement MCMC sampling scheme with fixed $\tau$ and the most recent values of $(\boldsymbol{\pi}, \boldsymbol{\sigma^2})$, for obtaining Bayesian posterior estimates of $(\boldsymbol{\beta}, \boldsymbol{\gamma})$;

(iv) M-step: Solve for the maximum likelihood estimates (MLEs) of $(\boldsymbol{\pi}, \boldsymbol{\sigma^2})$ by maximizing the expected log-posterior-likelihood functions [1], in which $\boldsymbol{\beta}$ and the expectation of $\boldsymbol{\gamma}$ are

approximated by their most recent Bayesian posterior estimates (from the E-step);

(v) Repeat the EM-steps (iii and iv) for a few times (e.g., 5 interations) until the MLEs of $(\boldsymbol{\pi}, \boldsymbol{\sigma^2})$ converge.

### 1.3.1 Setup Initial Values

By default, bfGWAS chooses $a_q$ and $b_q$ such that the mean of the Beta distribution (prior for $\pi_q$) $\frac{a_q}{a_q+b_q} = 1 \times 10^{-6}$ and $a_q + b_q = m_q = \sum_{i=1,j=q}^{p} A_{ij}$ (the total number of variants with annotation $q$); sets $k_1 = k_2 = 0.1$ to induce non-informative priors on $\sigma_q^2$; takes initial values $\pi_q = 1 \times 10^{-6}$ to imply a sparse model, $\sigma_q^2 = 10$ to start with a relative large effect-size variance for all causal variants.

The software bfGWAS fixes $\tau$ as the phenotype variance (1 with standardized phenotype vector $\boldsymbol{y}$), which is equivalent to assuming no phenotype variance explained by the genetic variants (this is true for most of the genome blocks because true risk loci usually locate in a few genome blocks). However, when there is one or multiple true associations locate in one genome block, bfGWAS has good power to identify these associations.

### 1.3.2 MCMC Sampling Scheme

With fixed values for $(\boldsymbol{\pi}, \boldsymbol{\sigma^2}, \tau)$, MCMC sampling is implemented independently per genome block. Take block $k$ for an example, the goal is to obtain the posterior estimates for $\boldsymbol{\gamma_k}$ and $\boldsymbol{\beta_k}$, which are given by the corresponding posterior sample means.

### 1.3.3 EM Update

The conditional posterior density function (posterior likelihood) of $\boldsymbol{\sigma^2}$ is

$$P(\boldsymbol{\sigma^2}|\boldsymbol{\beta}, \boldsymbol{\gamma}, \tau) \propto P(\boldsymbol{\beta}|\boldsymbol{\gamma}, \boldsymbol{\sigma^2}, \tau) P(\boldsymbol{\sigma^2}). \tag{3}$$

Since the posterior distributions of $\{\sigma_q^2;\, q = 1, \ldots, Q\}$ are disjoint, the expected log-posterior-likelihood function for each $\sigma_q^2$ is

$$l_{\sigma_q^2} = \sum_{j_q=1}^{m_q} \left[ \widehat{\gamma_{j_q}} \left( \frac{1}{2} ln \left( \frac{\tau}{\sigma_q^2} \right) - \frac{\tau \widehat{\beta_{j_q}}^2}{2\sigma_q^2} \right) \right] + (k_3 + 1) ln \left( \frac{1}{\sigma_q^2} \right) - \frac{k_4}{\sigma_q^2} + C, \tag{4}$$

where $\{\widehat{\gamma_{j_q}}, \widehat{\beta_{j_q}};\ j_q = 1, \ldots, n_q\}$ are the Bayesian estimates for variants of annotation $q$, and $m_q$ is the total number of variants with annotation $q$. The MLE of $\sigma_q^2$ can be obtained as

$$\widehat{\sigma_q^2} = \frac{\tau \sum_{j_q=1}^{m_q} (\widehat{\gamma_{j_q}} \widehat{\beta_{j_q}^2}) + 2k_4}{\sum_{j_q=1}^{m_q} \widehat{\gamma_{j_q}} + 2(k_3+1)}.$$

The conditional posterior density function (posterior likelihood) of $\boldsymbol{\pi}$ is

$$P(\boldsymbol{\pi}|\boldsymbol{\gamma}) \propto P(\boldsymbol{\gamma}|\boldsymbol{\pi}) P(\boldsymbol{\pi}). \tag{5}$$

Similarly, since the posterior distributions of $\{\pi_q;\ q = 1, \ldots, Q\}$ are also disjoint, the expected log-posterior-likelihood function for $\pi_q$ is given by

$$l_{\pi_q} = \sum_{j_q=1}^{m_q} \left[\widehat{\gamma_{j_q}} ln(\pi_q) + (1 - \widehat{\gamma_{j_q}}) ln(1-\pi_q)\right] + (a_q - 1) ln(\pi_q) + (b_q - 1) ln(1-\pi_q) + C, \tag{6}$$

and the MLE for $\pi_q$ is obtained as

$$\widehat{\pi_q} = \frac{\sum_{j_q=1}^{m_q} \widehat{\gamma_{j_q}} + a_q - 1}{m_q + a_q + b_q - 2}.$$

### 1.3.4 Construct Confidence Intervals by Fisher Information

By the asymptotic-normality of MLE, the MLEs $\widehat{\boldsymbol{\sigma^2}}$ and $\widehat{\boldsymbol{\pi}}$ are converging to Multivariate Normal (MVN) distributions as $n \to \infty$,

$$\widehat{\boldsymbol{\sigma^2}} \longrightarrow MVN(\boldsymbol{\sigma_*^2}, \boldsymbol{I_{\sigma^2}}(\widehat{\boldsymbol{\sigma^2}})^{-1}),\ \widehat{\boldsymbol{\pi}} \longrightarrow MVN(\boldsymbol{\pi_*}, \boldsymbol{I_{\pi}}(\widehat{\boldsymbol{\pi}})^{-1}), \tag{7}$$

where $\boldsymbol{\sigma_*^2}$ and $\boldsymbol{\pi}_*$ are the true parameter values; $\boldsymbol{I_{\sigma^2}}(\widehat{\boldsymbol{\sigma^2}}) \approx -\frac{\partial^2 l(\boldsymbol{\sigma^2})}{\partial \boldsymbol{\sigma^2} (\partial \boldsymbol{\sigma^2})^T}|_{\widehat{\boldsymbol{\sigma^2}}}$; and $\boldsymbol{I_{\pi}}(\widehat{\boldsymbol{\pi}}) \approx -\frac{\partial^2 l(\boldsymbol{\pi})}{\partial \boldsymbol{\pi} \partial \boldsymbol{\pi}^T}|_{\widehat{\boldsymbol{\pi}}}$. The Fisher informations of $\sigma_q^2, \pi_q$ are given by

$$I(\sigma_q^2) = \frac{dl_{\sigma_q^2}}{d^2\sigma_q^2} = \frac{1}{(\sigma_q^2)^2} \left( \sum_{j_q=1}^{m_q} \widehat{\gamma_{j_q}} (\tau - 0.5) - (k_3+1) + \frac{2k_4}{\sigma_q^2} \right),$$

$$I(\pi_q) = \frac{dl_{\pi_q}}{d^2\pi_q} = \frac{\sum_{j_q=1}^{m_q} \widehat{\gamma_{j_q}} + a_q - 1}{\pi_q^2} + \frac{n_q - \sum_{j_q=1}^{m_q} \widehat{\gamma_{j_q}} + b_q - 1}{(1-\pi_q)^2}.$$

The $(1-\alpha)\%$ confidence intervals of $\sigma_q^2, \pi_q$ can be constructed by

$$\widehat{\sigma_q^2} \pm Z_{\alpha/2} \sqrt{I(\widehat{\sigma_q^2})^{-1}},\ \widehat{\pi_q} \pm Z_{\alpha/2} \sqrt{I(\widehat{\pi_q})^{-1}}, \tag{8}$$

where $Z_{\alpha/2}$ is the upper $\alpha/2$ quantile of the standard normal distribution.

## 1.4 Missing Data

### 1.4.1 Missing Genotypes

Variants with missing percentage larger than the threshold given by `-miss [num]` or default 0.05 are excluded from analysis. Otherwise, the missing genotypes will be replaced by sample mean.

### 1.4.2 Missing Phenotypes

Samples with missing phenotypes (`NA`) will be excluded from the analysis.

### 1.4.3 Missing Annotations

Missing annotations (empty or `NA` in the annotation file) are suggested to be treated as one category.

## 2 Input File Formats

To conduct a Bayesian GWAS with functional annotations, bfGWAS requires five input files: genotypes, phenotypes, variant annotations, annotation classification codes, initial parameter values.

### 2.1 Genotype file

The genotype file has to be of one of the following formats:

- VCF, text file zipped by `gzip` (file names ending with `.vcf.gz`), e.g., `chr1_seg1.vcf.gz` (`http://samtools.github.io/hts-specs/VCFv4.1.pdf`).
- PLINK binary PED file, accompanied with bim (variant information) and fam (genotype information) files, e.g., `chr1_seg1.bed`, `chr1_seg1.fam`, `chr1_seg1.bim` (`http://pngu.mgh.harvard.edu/~purcell/plink/data.shtml#bed`). The bim file has six columns: chromosome (1-22, X, Y or 0 if unplaced), variant ID, genetic distance (margans), base-pair position (bp units), reference and alternative alleles.
- Genotype text file with tab-delimited columns, zipped by `gzip` (file names ending with `.geno.gz`), e.g., `chr1_seg1.geno.gz`, :

  | ID | CHROM | POS | REF | ALT | HG00096 |
  |---|---|---|---|---|---|
  | rs192352289 | 1 | 195000420 | C | A | 1.20 |
  | rs189909013 | 1 | 195000479 | C | T | 1.20 |
  | rs74350489 | 1 | 195000492 | G | T | 1.20 |
  | rs12402744 | 1 | 195000519 | T | C | 1.20 |

  where the first row is of headers: variant id, chromosome number, base pair position, reference allele, alternative allele, and sample ids starting on the sixth column. Each row represents one variate. Note that the header for the first column has to be `ID` to be recognized by the software, and genotypes of each sample (starting from the sixth column) are either dosage data in $[0, 2]$ or minor allele numbers in $\{0, 1, 2\}$.

If the variant ID in the genotype files are missing, i.e. either empty or denoted by ".", the variant ID will be set as "CHROM:POS:REF:ALT" by the software.

### 2.2 Phenotype file

The phenotype file accompanied with the vcf/genotype files (text file either zipped by gzip or not) contains two columns separated by either spaces or tab: The first column is of sample ids as in the VCF genotype file or genotype text file. The second column is of quantitative phenotypes

(cases can be represented by 1s and controls by 0s). Notice that only numeric values are allowed for the second column and characters will not be recognized by the software. Missing phenotype information should be denoted as NA. Phenotype data will be matched with genotypes in the genotype files by sample ids (i.e., sample orders in the phenotype file can be random). An example phenotype file (`pheno.txt.gz`) with five individuals and one phenotype is as follows:

```
HG00096 1
HG00097 1
HG00099 0
HG00100 1
HG00101 0
```

If genotype information is provided by a bed file, the phenotype information needs to be contained in the sixth column of the corresponding fam file (e.g., `chr1_seg1.fam`). Similarly, only quantitative values will be recognized by the software (only $-9$ and `NA` will be treated as missing values; 0 will be loaded as it is). The first five columns of a fam file are "FamilyID, IndividualID, PaternalID, MaternalID, Sex", and the samples should match the order of genotypes in the bed file. An example fam file looks like (0 denotes missing values for the first five columns):

```
HG00096  HG00096  0 0 0  1
HG00097  HG00097  0 0 0  1
HG00099  HG00099  0 0 0  0
HG00100  HG00100  0 0 0  1
HG00101  HG00101  0 0 0  0
```

We recommend first regressing covariants out from the original phenotypes, and then provide bfGWAS the corrected phenotypes (i.e., residuals from the regression model with covariates).

### 2.3 Annotation file

The annotation file has four columns separated by either a white space or tab, representing variant ids, chromosome numbers, base pair positions, and annotations (represented by characters) in order. The first row is for column names, while the remain rows are for variants in exactly the same order as in the genotype file (bfGWAS will NOT double check this). Note that the annotations could be missing, but the variants shall still exist in the annotation file with empty annotations.

An example annotation file (`Anno_chr1_seg1.gz`) with four SNPs is as follows:

```
ID              CHROM   POS             ANNO
rs2360961       1       199000277       intergenic
rs114296307     1       199000286
rs186415274     1       199000350       NA
rs4259690       1       199000408       intergenic
```

In this example, the second and the third variants have no annotations (could be either blank or denoted by `NA`).

Such an annotation file can be easily prepared by using `awk`. For example, assume there is a file of annotated variants (`variant_anno.txt`, separated by tabs) like:

```
ID              ANNO
rs2360961       intergenic
rs114296307
rs186415274     NA
rs4259690       intergenic
```

The following command will create the annotation file corresponding to a genotype file in vcf format (replace `gzip -dc chr1_seg1.vcf.gz` by `cat chr1_seg1.vcf` if your vcf file is not gzipped):

```
awk -F '[\t]' 'FNR==NR { a[$1]=$2; next;} \
NF>2{print $3  "\t"  $1 "\t"  $2 "\t" a[$3]}' \
<(cat variant_anno.txt) <(gzip -dc chr1_seg1.vcf.gz | cut -f 1-3) | gzip \
> Anno_chr1_seg1.gz
```

The following command will create the annotation file corresponding to a bim file:

```
awk -F '[\t]' 'FNR==NR { a[$1]=$2; next;} \
{print $2  "\t"  $1 "\t"  $4 "\t" a[$2]}' \
<(cat variant_anno.txt) <(cat chr1_seg1.bim) | gzip > Anno_chr1_seg1.gz
```

The following command will create the annotation file corresponding to a genotype text file (replace `gzip -dc chr1_seg1.geno.gz` by `cat chr1_seg1.geno` if your genotype file is not gzipped):

```
awk -F '[\t]' 'FNR==NR { a[$1]=$2; next;} \
{print $1  "\t"  $2 "\t"  $3 "\t" a[$1]}' \
<(cat variant_anno.txt) <(gzip -dc chr1_seg1.geno.gz | cut -f 1-3) | gzip \
> Anno_chr1_seg1.gz
```

All annotations will be matched to genotypes by variant IDs. If the genotype file has missing variant IDs (i.e., either empty or denoted by "." in the ID field), the variant ID in the annotation file should be set as "CHROM:POS:REF:ALT" to find the genotype match.

If annotation file is missing, bfGWAS will treat all variants as if they were of the same category,

.

### 2.4 Classification code file

The file of annotation classification codes (tab-delimited txt file) contains information about how to classify the variant annotations. The first row has to be of the format `#n_type Q`, where `Q` should be replaced by the total number of categories. Starting from the second row, the first column contains all unique annotation names, while the second column denotes the categories by consecutive integers starting from 0 (one integer per category).

In the following example (`AnnoCode6.txt`), although the variants are annotated with 7 unique annotations (as in the first column), we group them into 6 categories denoted by `0, ..., 5` respectively. Note that missing annotation or `NA` annotation need to be assigned a category for analysis as well. Here, we grouped it into the `intergenic` category.

```
#n_type 6
coding 0
UTR 1
promoter 2
DHS 3
intronic 4
intergenic 5
NA 5
```

This file of annotation classification codes allows one to conduct analysis with different partitions of the annotations, without the necessity of generating a different set of annotation files. Note that this file is required if you want to analyze your data in multiple annotation categories. Otherwise, all variants will be treated as if they were of the same category.

### 2.5 Initial parameter file

The initial parameter file (tab-delimited txt file) contains the initial values for $\pi_q, \sigma_q^2$, where the first column contains values for $\pi_q$, and the second column contains values for $\sigma_q^2$ (in the same order for categories $0, 1, 2, \cdots$ as coded by the annotation code file).

In the following example (`InitValues6.txt`), the first row contains column names, while the remain rows contains initial values of $\pi_q, \sigma_q^2$, for $q = 0, ..., 5$.

```
pi      sigma2
1.0e-6  10.0
1.0e-6  10.0
1.0e-6  10.0
1.0e-6  10.0
1.0e-6  10.0
1.0e-6  10.0
```

Note that any unspecified category-specific parameters will be set at default values: $\pi_q = 1.0 \times 10^{-6}$, $\sigma_q^2 = 10.0$.

## 3 Software bfGWAS

The software bfGWAS is freely available at Github (`https://github.com/yjingj/bfGWAS`). The whole software has three parts:

(i) the executable file (`./bin/Estep_mcmc`) generated by C++ source code for running the MCMC algorithm per genome block. Users may either use the en-closured executable file directly (compiled for Linux system), or revise the en-closured `Makefile` to compile their own version with the C++ source codes (`./src/`). Note that standard C/C++ compiler such as GNU gcc, libraries GSL (`http://www.gnu.org/s/gsl/`), LAPACK (`http://netlib.org/lapack/`) are required.

(ii) the R script (`./bin/Mstep.r`) for updating category specific parameters ($\pi_q, \sigma_q^2$) by the EM algorithm.

(iii) the Perl script (`./bin/gen_mkf.pl`) to generate an analysis Makefile for handling parallel MCMC jobs and EM updating jobs. Users can also edit the Perl script according to their needs.

The idea is creating job scripts for running the `Estep_mcmc` per genome-block and the Rscript in the analysis Makefile, and then using the analysis Makefile to control parallel computing process and the EM steps. Analysis results will be generated after successfully run the analysis Makefile.

### 3.1 Implement bfGWAS

To implement bfGWAS to conduct Bayesian GWAS with functional annotations, first run the Perl script (`gen_mkf.pl`) to generate an analysis Makefile, and then run the analysis Makefile. Example usage of `gen_mkf.pl` is given as follows:

```
${tool_dir}/bin/gen_mkf.pl \
-w ${wkdir} --Estep ${tool_dir}/bin/Estep_mcmc \
--ad ${data_dir}/annos --geno vcf --ac ${data_dir}/AnnoCode6.txt  \
--gd ${data_dir}/vcfs --pheno ${data_dir}/phenoAMD_1KG.txt \
--hyp ${data_dir}/InitValues6.txt -f ${data_dir}/fileheads_4region.txt \
--rs ${tool_dir}/bin/Mstep.r --maf 0.005 --smax 10 \
-b 50000 -N 50000 --NL 50000 --em 5 -j testjob -l slurm --mem 3000 \
--mf ${wkdir}/Test_bfGWAS.mk
```

Details about the options for the Perl script can be found by running `gen_mkf.pl -h`.

Then run the generated analysis Makefile as

```
make -k -C ${wkdir} -f ${wkdir}/Test_bfGWAS.mk -j 4 \
> ${wkdir}/make.output 2> ${wkdir}/make.err  &
```

where `-C` specify the running directory, `-f` specify the analysis Makefile, and `-j` specify the number of parallel jobs to be run.

If a job is completed successfully, a corresponding `OK` file will be generated. Even if running the analysis Makefile fails at some point, users can simply run the Makefile again, without repeating the accomplished jobs. The Makefile will automatically detect unaccomplished jobs by looking for the corresponding `OK` files.

If the users want to run the analysis again, simply erase all `OK` files generated from last time by

```
make -f ${wkdir}/Test_bfGWAS.mk clean
```

and then run the Makefile again. If the users want to rerun one of the jobs, simply erase the corresponding `OK` file and run the Makefile again.

More details about Makefiles are referred to `https://www.gnu.org/software/make/manual/html_node/Makefiles.html`.

### 3.2 Memory Usage

Without specifying in-memory compression, bfGWAS usually takes about $3,000$MB memory to analyze a genome block with $10,000$ variants, $30,000$ samples, and $50,000$ MCMC iterations; and about 10GB memory to analyze a genome block with $500,000$ variants, $30,000$ samples, and $50,000$ MCMC iterations. The memory usage approximately increases in the order of $O(npM)$, with sample size $n$, variant number $p$, and MCMC iterations $M$.

Users are suggested to test on one genome block to obtain an approximate memory usage, and then set the `--mem` option (for `gen_mkf.pl`) at a sufficient value.

### 3.3 Output Files

By running the analysis Makefile, directories of `OUT` and `Eoutput` will be generated to save all outputs. The outputs from running `Estep_mcmc` per genome-block are saved under the directory `OUT`, and the combined outputs from EM iterations are saved under `Eoutput`. The final analysis results for all variants are saved in the `paramtemp${iter}.txt` file; the hyper parameter values are saved in the `EM_result.txt` file; the hyper parameter values per genome-block are saved in the `hyptemp${iter}.txt`; the logs of the MCMC process are saved in the `log${iter}.txt` file. We mainly interested with the results of the last EM-iteration, in the `EM_result.txt` (last row) and `paramtemp${last_iter}.txt` files.

We provide analyzing R functions in `bin/R_funcs.r`, and an example Rscript `AnalyzeResults/Analysis.r` for analyzing the bfGWAS outputs of the example GWAS in Section 3.4.

### 3.4 Artificial Example GWAS Dataset

We used genotypes of 4 real risk loci of age-related mucular degeneration AMD from the 1000 Genome (1KG) project (Phase 3), and 15 AMD hits in these 4 loci, to simulate 1237 AMD cases and 1267 controls. We provide simulated phenotype file at `1KG_example/ExData/phenoAMD_1KG.txt`; genotype vcf files at `1KG_example/ExData/vcfs/`; and corresponding annotation files at `1KG_example/ExData/annos/`.

As results of applying the bfGWAS as in Section 3.1, the raw outputs are in the `1KG_example/Test_Wkdir` folder; analysis Rscripts and plots are in the `1KG_example/AnalyzeResults` folder.

Test script is available at `1KG_example/Test_Wkdir/test_script.txt`.

## 4 Options for bfGWAS

### 4.1 Options for the Perl Script

Note that all options with a single letter are specified by starting with single dash `-`, and all options with more than one letter are specified by starting with double dashes `--`.

File I/O Related Options

- `-w [dir]` specify work directory;
- `-t [dir]` specify directory of the directory for the executable C++ software `Estep_mcmc`;
- `--geno [filetype]` specify genotype file format: either "vcf", "genotxt", or "bed";
- `--gd [dir]` specify genotype file directory;
- `--ad [dir]` specify annotation file directory;
- `--ac [filename]` specify annotation classification code file;
- `--pheno [filename]` specify phenotype file;
- `--hyp [filename]}` specify the file with initial hyper parameter values;
- `-f [filename]` specify the file with a list of fileheads for genotype files;
- `-G [datatype]` specify the genotype data type in vcf files: either "GT" for 0/0, 0/1, 1/1, or "EC" for dosage data.

EM-MCMC Related Options

- `--maf []` specify maf threshold, default 0.5%;

- `--pp []` specify prior for the causal probability per annotation group, default 1e-6;
- `--abgamma []` specify prior for the effect size variance per annotation group, default 0.1;
- `--win []` specify the window size of neighborhood variants: default 100;
- `--em []` specify number of EM iterations: default 5;
- `-b []` specify number of burnins for the MCMC algorithm: default 50,000;
- `-N []` specify number of iterations for the MCMC algorithm: default 50,000;
- `--NL []` specify number of iterations for the MCMC algorithm in the last EM iteration: default 50,000;
- `--initype []` specify the initial model for the MCMC algorithm: (1:start with top signal), (2: start with genome-wide significant signals), or default (3: stepwise selected signals)
- `--rv []` specify fixed residual variance percentage: default 1 (recommended);
- `--smin []` specify minimum number of variates in the linear model in the MCMC iterations: default 0;
- `--smax []` specify maximum number of variates in the linear model in the MCMC iterations: default 5;

Computation Related Options

- `--mf [dir]` specify directory for the Makefile;
- `-c []` specify whether compress genotype data (1) or not (0): default 0 (do not compress);
- `--mem []` specify the memory usage for MCMC jobs, in the unit of MB: default 3,000 (MB);
- `--time []` specify the running time for MCMC jobs: default 24hr;
- `-l []` specify how jobs will be submitted: either default "slurm", "mosix", "local";
- `-j []` specify job name;
- `--xnode []` specify computation nodes to be excluded
- `--wnode []` specify computation nodes to be used
- `--nice []` SLURM option for scheduling priority
- `--part []` SLURM option for specifying partition

### 4.2 Executable C++ Software

The executable C++ part `Estep_mcmc` can also be run alone to:

- Analyze a single genotype file;
- Generate kinship matrix;
- Conduct single variant analysis based on linear models.

Manual is available by `./Estep_mcmc -h`.

### 4.3 Variant filters

Here are some common variant filters implemented in `Estep_mcmc`.

- Polymorphism. Non-polymorphic variants will excluded from the analysis.
- Missing rate. By default, SNPs with missing rate above 5% will not be included in the analysis. Use "-miss [num]" to change. For example, "-miss 0.1" changes the threshold to 10%.
- Minor allele frequency. By default, variants with minor allele frequency below 0.5% will not be included in the analysis. Use "-maf [num]" to change. For example, "-maf 0.05" changes the threshold to 5%.
- Hardy-Weinberg equilibrium. Use "-hwe [num]" to specify. By default, variants with Hardy-Weinberg $p$ values below 0.001 will not be included in the analysis. For example, "-hwe 0.05" will change the significance level to 0.05.

Calculations of the above filtering thresholds are based on analyzed individuals (i.e. individuals with no missing phenotypes and no missing covariates). Therefore, if all individuals have missing phenotypes, no variant will be analyzed and the output matrix will be full of "NA"s.

## References


[1] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):pp. 1–38, 1977.

[2] Edward I. George and Robert E. McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

[3] Yongtao Guan and Matthew Stephens. Bayesian variable selection regression for genome-wide association studies and other large-scale problems. *Ann. Appl. Stat.*, 5(3):1780–1815, 09 2011.

[4] Xiang Zhou, Peter Carbonetto, and Matthew Stephens. Polygenic modeling with bayesian sparse linear mixed models. *PLoS Genet*, 9(2):e1003264, 02 2013.

[5] Xiang Zhou and Matthew Stephens. Genome-wide efficient mixed-model analysis for association studies. *Nat Genet*, 44(7):821–824, 07 2012.